

## **IN THE DRAWINGS**

The Examiner has requested correction to Figures 7A, 7B, 8A, 8B and 10 for minor informalities. Applicant is submitting replacement drawings for Figures 7A, 7B, 8A and 8B. The specification has been amended to correct the informality of Figure 10. No new matter has been added.

## **IN THE SPECIFICATION**

**Please replace paragraph [0028] with the following:**

[0028] The VMM 112 presents to other software (i.e., “guest” software) the abstraction of one or more virtual machines (VMs), which may provide the same or different abstractions to the various guests. **Figure 1** shows three VMs, 130, 140 and 150. The guest software running on each VM may include a guest OS such as a guest OS 151, 160 or 170 and various guest software applications 152, 162 and 172. The guest OSs 151, 160 or 170 expect to access physical resources (e.g., processor registers, memory and I/O (Input/Output) devices, within a corresponding VM (e.g., VM 130, 140 or 150) on which the guest OS is running and to perform other functions. For example, the guest OS expects to have access to all registers, caches, structures, I/O (Input/Output) devices, memory and the like, according to the architecture of the processor and platform presented in the VM. Further, each guest OS expects to handle various events such as exceptions, interrupts, and platform events (e.g., initialization (INIT) and system management interrupts (SMIs)).

**Please replace paragraph [0064] with the following:**

[0064] If the VMCS is in the cleared state, processing logic performs a variety of checks of the state in the processor and VMCS (processing block 706). For example, the PG (Paging) bit in the CR0 (Control Register 0) register may be required to be set following completion of the

VM entry instruction. If these checks indicate errors (decision box 708), then processing logic fails the VM launch instruction, setting error codes and returning control to the VMM (processing block 720). Otherwise, if the verification checks succeed, processing logic stores VMM state to the VMCS and loads guest state from the VMCS into the processor (processing block 710). If there are problems detected during the storing or loading of state (decision box 712), processing logic fails the VM launch instruction (processing block 718). If the storing of VMM state and loading of guest state completes without error, processing logic marks the VMCS as being in the "launched" state (processing block 714) and then begins executing the guest (processing block 716).

**Please replace paragraph [0081] with the following paragraph:**

[0081] Accordingly, processor operation in response to the VM resume instruction is simplified. In particular, some necessity for validation of state in VMCS is removed because the processor can guarantee that state stored in processor-managed resources has not been "scribbled" by software. For example, a processor may be required to verify that the CR0 PG (Control Register 0 Paging) bit is set prior to entry to a guest. Assume that the CR0 (Control Register 0) field of guest state is maintained in processor-managed resources following the first entry to a guest. If, prior to a second entry to a guest, the CR0 (Control Register 0) field has not been accessed (e.g., through the execution of the VMCS write instruction), then the processor does not need to perform this validation check during the VM resume instruction (because the value in the processor-managed resource could not have been changed).

**Please replace paragraph [0091] with the following paragraph:**

**[0091]** Next, processing logic determines how processor-managed resources associated with the VM are to be utilized based on the behavior of the VMM (processing block 1004). Exemplary uses of processor-managed resources include allocation or deallocation of certain processor-managed resources, invalidation of data stored in processor-managed resources, verification of data stored in processor-managed resources, loading of data into processor-managed resources, etc.